I J C T A, 9(17) 2016, pp. 8489-8497

Non-uniform Slant Correction using Generalized Projections

A. M. Hafiz[*] and G. M. Bhat[*]

ABSTRACT

Slant Correction is an important component of an Optical Character Recognition (OCR) system. The stronger the slant correction algorithm, the better the OCR system's performance. In this work, a non-uniform slant estimation algorithm is presented. The algorithm uses the generalized projections of the word image. The algorithm is simple and robust. To evaluate the efficacy of the algorithm, images from IAM database were classified using a Support Vector Machine-based recognition engine. The performance of the proposed technique has been compared to other contemporary techniques. The results show attest the efficacy of the proposed technique.

Keywords: Non-uniform slant correction, OCR, IAM Database, Artificial Intelligence

I. INTRODUCTION

Slant correction is an important pre-processing stage of Optical Character Recognition (OCR) systems, e.g. Hidden Markov Model (HMM) based OCR systems [1, 2] where slant of text plays an important role. Its efficiency directly contributes to the overall accuracy of the OCR system. Both uniform and non-uniform slant-correction techniques have been developed [3-11]. Uniform slant-correction techniques are unable to deal with variable slant within the text to be de-slanted. The non-uniform slant-correction approaches usually use complex slant-estimation techniques like Dynamic Programming (DP) [3]. In this work, a non-uniform slant-correction procedure has been proposed which is simple and robust. This approach is based on image-processing of the Generalized Projections [12] of handwritten text images. The IAM Database of Handwritten Words [13] was used for experimentation. A SVM-based OCR engine was used to classify the slant-corrected text images. The performance of the Generalized Projection -based slant-correction has been compared to other slant-correction techniques. The results of experimentation suggest the efficacy and robustness of the proposed technique.

II. GENERALIZED PROJECTIONS

The radial histogram [14] of the binary image is obtained by taking the radial count around the text. The radial histogram r_φ at an angle φ is the sum of black pixels on a rad that starts from the center of the graphical object matrix (the N^{th} row and the N^{th} column element) and ends up at the border of the matrix, while forming an angle with the horizontal axis:

$$r_\varphi = \Sigma_{i=1}^{N} I(\| N - i \sin\varphi \|, \| N + i\cos\varphi \|) \quad \varphi = 1, ..., 360 \tag{1}$$

The Generalized Projection (GP) [12] is also a histogram. It is an array of fore-ground pixel count along each successive perpendicular plane of the histogram line, which itself rotates along the center of the graphical object. Given a N×N image, x_c and y_c are given by:

[*] Department of Electronics and Communication Engineering, Kashmir University Institute of Technology, University of Kashmir, Srinagar, J&K, India, *E-mail: mueedhafiz@yahoo.com*

[**] University Science Instrumentation Center, University of Kashmir, Srinagar, J&K, India, *E-mail: drghmbhat@gmail.com*

$$\left.\begin{aligned} x_c\left(i+\tfrac{N}{2}+1\right) &= i+\tfrac{N}{2}+1 && i=-\tfrac{N}{2},\ldots,-1 \\ y_c\left(i+\tfrac{N}{2}+1\right) &= floor(2\cdot N+|i|\cdot\tan\theta) && i=-\tfrac{N}{2},\ldots,-1 \end{aligned}\right\} \quad (2)$$

and

$$\left.\begin{aligned} x_c\left(i+\frac{N}{2}\right) &= i+\frac{N}{2} && i=1,\ldots,\frac{N}{2} \\ y_c\left(i+\frac{N}{2}\right) &= floor(2\cdot N-|i|\cdot\tan\theta && i=1,\ldots,\frac{N}{2} \end{aligned}\right\} \quad (3)$$

and the image vertically runs from –N/2, ..., N/2, and θ is the angle of histogram, then the GP component (rotated histogram) for an angle θ, $\boldsymbol{R}_\theta$ is given by:

$$R_\theta(i)=\Sigma_{j=1}^{N}\ I(x_c(j),y_c(j)) \qquad i=1,\ldots,N \quad (4)$$

Figure 1 shows a word image and its GP.

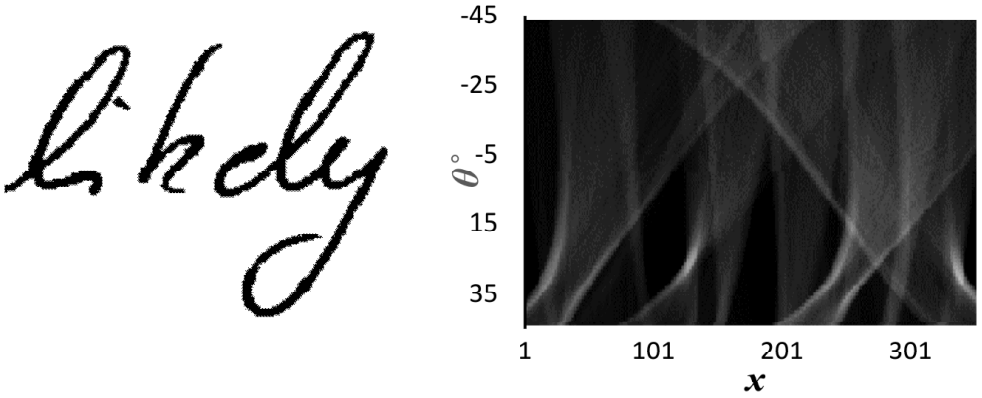

Figure 1: A word image (left), and it Generalized Projection (right)

Note the hotspots in the GP shown in Figure 1 at an angle of 30°, which is the approximate text slant.

III. SLANT DETECTION

GP has been previously used for non-uniform slant-estimation [4] with Dynamic Programming [3] has been used for non-uniform slant estimation. Dynamic Programming is complex and computationally expensive. The proposed approach is image-processing based. Here, GP of the word image is treated as a grayscale image. Next, it is binarized by normalization and then thresholding using a threshold β given by:

$$\beta = 0.4 - e^{-(-22\gamma+1)} \quad (5)$$

Where γ is calculated as:

$$\gamma=\frac{\Sigma_{i=1}^{w}\Sigma_{j=1}^{h}I(i,j)}{w\ h\ \max(I)} \quad (6)$$

Here I is GP of the Grayscale Image, w is the width of I, h is the height of I. Equation 5 is the curve-fitting approximation function for values of β which gave optimum fault line detection. The curve-fitting is illustrated in Figure 2.

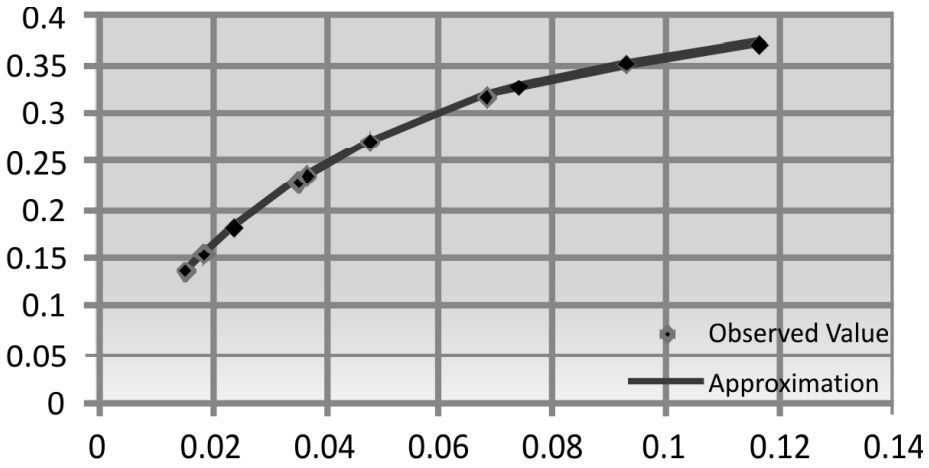

Figure 2: Curve-Fitting for

Once the binary GP is obtained, the resulting white hotspot blobs in the image are used to obtain the non-uniform slant lines. Figure 3 shows some word images (grey) with their detected slant-lines (black).

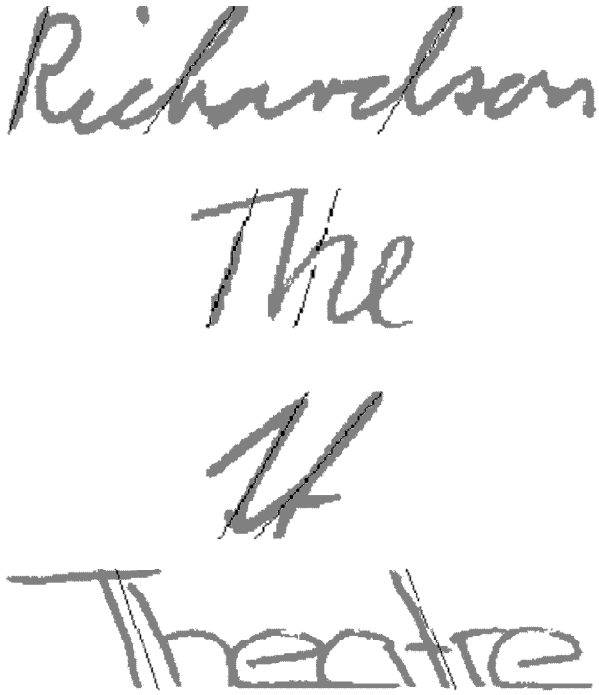

Figure 3: Non-uniform Slant Lines (black) detected

For slant correction, two slant lines with θ = 0° are added at x=1 and x=[width of image].

IV. SLANT CORRECTION

Interpolation Technique [6] was used for interpolation, where the discrete slant values are converted into continuous slant values for all x-values of the word image. The slant at a position x between two slant lines x_k and x_{k+1}, i.e. for $x_k \le x \le x_{k+1}$, is given in Equation 7.

$$\theta(x) = \frac{(x - x_k)^2 L_{k+1} \theta_{k+1} + (x_{k+1} - x)^2 L_k \theta_k}{(x - x_k)^2 L_{k+1} + (x_{k+1} - x)^2 L_k} \tag{7}$$

$$k = 0, \ldots, K$$

where x_k, θ_k and L_k are the horizontal position, the estimated angle and the length of the k^{th} slant line, respectively. K is the number of slant lines. A second order interpolation equation has been used which gives optimum results. The new values of slant-corrected x-coordinates and y-coordinates, for all pixels are obtained from Equations 9-10 and 9-11.

$$x_{new} = x_{old} - y_{old} \tan\left(\theta(x_{old})\right) \tag{8}$$

$$y_{new} = y_{old} \tag{9}$$

Two problems were encountered in the slant correction approach. First was that when two slant lines overlapped, it led to distorted corrected image. This problem was solved by first predicting overlap, and if present, the average of x-coordinates and slant-angles for each such pair was taken. The second problem was more complicated. It arose from high variations in successive slant-angles for two neighboring slant lines, resulting in distorted corrected images. This problem was earlier addressed [6] by using a multistep approximation procedure. In this paper, an efficient, simpler and faster approach is used, which is as follows. After slant correcting the word image, all successive four neighboring horizontal pixels were compared. If all four were black, a counter initialized to zero, was incremented. After whole of the image was scanned, and the counter was fully updated, Equation 10 was used to determine an indicator m_1.

$$m_1 = \frac{m}{hw} \tag{10}$$

If the indicator fell within the experimentally observed range given as:

$$0.0275 \le m_1 \le 0.0475$$

then the original slanted word image was slant corrected afresh using Equations 9 and 11.

$$x_{new} = x_{old} - y_{old} \tan\left(\frac{\theta(x_{old})}{47.5 m_1}\right) \tag{11}$$

The objective achieved by modifying Equations 8 to 11 for slant correction is to reduce all the slant-angles by a common factor so that the variations in slant (if high) get smoothened out. Figure 4. shows a word image, its slant corrected images and interpolated slant-angle profile. The slant lines are also shown (in black). Note the fast variations in slant-angle leading to distortions in the de-slanting process.

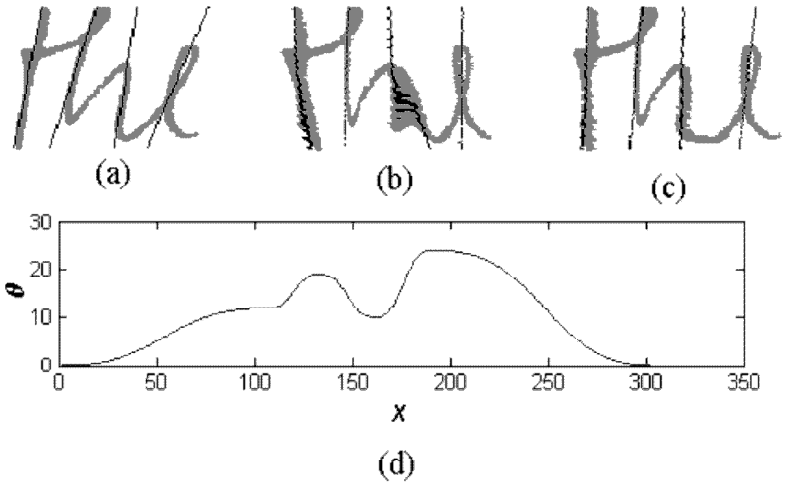

Figure 4: (a) Original slanted image, (b) Distorted de-slanted image, (c) De-slanted image obtained after proper slant-correction, (d) Interpolated slant-angle profile of the original word image

Shown in Figure 5 are some words and their corrected forms using the proposed approach.

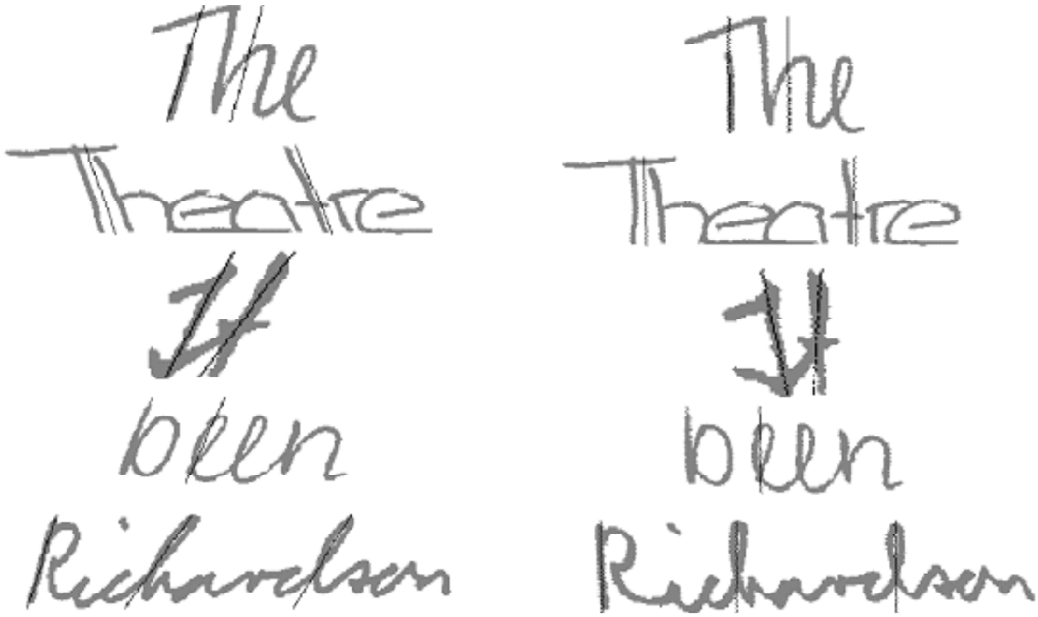

Figure 5: Some words (left), and their corrected forms (right).Slant lines are shown in black

V. RECOGNITION

5.1. Recognition

For demonstrating the efficacy of the proposed approach, a recognition engine was built for the slant-corrected samples. The recognition engine consists of the components discussed below.

5.1.1. Segmentation Component

A binary word-image vertical-histogram-based approach was used for segmentation.

5.1.2. Segment Combination

Segmentation is followed by segment combination. Given the nature of the word images in the dataset, all letters in the images generally would not be wider than three segments horizontally. Hence, a maximum of 3 segments were combined. Next, each image element of seg_comb{}, the segment combination array, was processed to present it to the classifier. This involved cropping the black pixels within a bounding box and resizing it to a 30x30 pixel size. For narrow letters like 'i' a vertical histogram was used to detect the actual width, and if found less than a certain threshold, the cropped image was buffered on both sides with background pixels, to maintain aspect-ratio. Also, if the length of the segment combination exceeded a certain limit, the element was dropped (to avoid useless images). Figure 6 shows generation of seg_comb{} array from seg_im{}, the raw segmented images array.

5.1.3. Classification

A SVM [15-17] Classifier, implemented using LIBSVM [18], and trained on the training set was used to classify each image element of seg_comb{}. The features used were Gradient Features [19-21]. The gradient components of the binary image were reduced to 8 gradient planes. Then each of the 30x30 pixel images of seg_comb{} was divided into 36 equally sized blocks i.e. 6 horizontal blocks and 6 vertical blocks. Then the gradient components along 8 respective planes were added for each block of the image. This gave an 8x36=288 element feature per image.

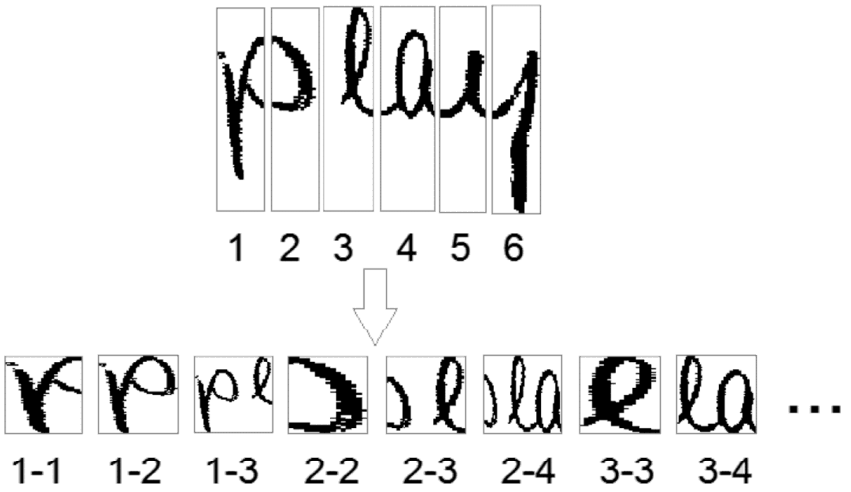

Figure 6: (Top) Elements of seg_im{} with indices, (Bottom) First seven elements of seg_comb{}, with indices range as per elements used from seg_im{}

Multiple successive prediction thresholds (determined experimentally) were used for the SVM Classifier. If no element of seg_comb{} was predicted using upper threshold=0.125, the threshold was lowered for element prediction successively (if needed) to 0.0625 or 0.0325 or 0.0125. If at lowest threshold no classification result was obtain, a null decision was passed to the next stage.

5.1.4. Post Classification Analysis

The order in which segments were combined was remembered. After classification, this order was used to weave the candidate strings from the recognized segments of seg_comb{}. More than one strings were obtained at this stage because of overlap between segmented images. The function used, involved a horizontal path finding function given a 3 row, n column, node tree, where n was number of recognized segments. There are 3 rows because for each element of seg_im{}, it can be used alone or it can be combined with next single or next two segment element of seg_im{}, giving an element of seg_comb{}. For the i^{th} column in the tree, the 1^{st} row node was i-i, second was i - (i+1), and the third was i - (i+2), where i=index of element of seg_im{}. This tree was converted into a binary graph where, every node o took value 1, if that combination was present in seg_comb{} or else it took value zero. A path was woven out of all nodes (1 per column at a time) if every nodes had value 1. Thus, there were 3^n paths (or strings) possible for the binary tree if all nodes have value 1.The actual number of paths for a binary tree is given by:

$$p = \Pi_{i=1}^{n} \Sigma_{j=1}^{3} o_{ij} \tag{12}$$

For each path computed as per the above algorithm, a string was generated by weaving nodes (each denoted by indices: ij, where i=index in seg_im{} and j=(i+1), or, j=(i+2), or, j=(i+3)). For string generation, a path (formed by non-zero nodes) to contain two nodes o_{ij} and o_{kl}, it was necessary that k>j. This ensured no overlap in segment combination. Each generated path gave a unique string as each recognized node had a letter value given by the classifier. Thus, a set of candidate strings was generated.

5.1.5. Similar String Generation

Once the set of candidate strings Set1 was generated, one more measure was taken to boost recognition. Those letters which looked somewhat similar (as per the data available) were substituted one by one in Set1 to generate an expanded set of string candidates Set2. That is all possible permutations were taken once more by a path-finding algorithm. The following were the letters and their duals:

r-T, r-t, l-e, e-l, e-c, c-e, o-D, D-o, D-b, b-D, M-m, m-M, g-q, q-g, t-r, t-f

5.1.6. String Comparison and Score Assignment

Each string of Set2 was compared to each word in the dataset dictionary (118 english words) using certain scoring rules. First, the current dictionary word under comparison, was broken down into constituent letters (length=1) and sub-strings (length from 2 to L-1, where L was number of letters in word). This data was queried for presence in the challenger string from Set2. The word as a whole was also queried for presence. The following rules were observed:

i) All data was queried in the same sequence in the challenger string from Set2, as was present in the current dictionary word.

ii) The longer the data string found the larger the increment in score for that dictionary word.

iii) If length of challenger string and dictionary word was same then more increment in score was given (upon presence) than when lengths were unequal.

iv) A penalty was deducted from score which was exponential with base two and power equal to absolute of difference of lengths.

Once scores were assigned to all strings in Set2, the following rules were applied to choose the winner:

i) If a maximum score was found for a certain prediction threshold (as discussed in Section 5.3), given number of elements in Set2 was greater than 3, then it was adjudged as the winner.

ii) If no clear majority was indicated or number of elements in Set2 was less than 3, Set2 was reconstituted for a lower prediction threshold. If a candidate with maximum score was present, then it was adjudged as the winner. Otherwise the threshold was lowered once more and Set2 was reconstituted. If failure was observed again, then threshold was repeatedly reduced and majority winner searched for.

iii) In case, prediction threshold was minimum (0.0125), then if a maximum score candidate was not present, then a majority maximum score holder was adjudged as the winner, else a non-decision was enumerated.

5.1.7. Ground Truth Comparison

Finally the winner of the recognition engine was compared with the ground truth value of the word image from the testing set.

VI. EXPERIMENTAL ANALYSIS

For demonstrating the efficacy of the non-uniform slant-correction approach, the corrected testing set (using the former) was compared with the corrected test sets using a both uniform and non-uniform slant-correction approaches. Figure 7 shows some words corrected using both uniform and non-uniform approaches. The proposed approach corrected the words better than the uniform approach.

The classification results for various approaches are given in Table 1.

Figure 7: Samples corrected using Uniform Approach (Left), and, Proposed Approach (Right).

Table 1
Classification Accuracy for various approaches

Slant-Correction Method	*Classification Accuracy (%)*
Non-uniform Approach (proposed)	91.46
Non-uniform Approach (DP-based) [5]	92.25
Uniform Approach [11]	78.94

The results show improvement in the recognition accuracy for the classifier when proposed non-uniform slant-correction approach is used, compared to when the uniform slant-correction approach is used. The results also suggest that the proposed approach is almost as effective as the popular DP-based approach, while being simpler than the latter.

VII. CONCLUSION

Non-uniform Slant Correction provides a way to de-slant words where there is in-word slant-variation. In this paper, a non-uniform slant correction approach has been proposed which is simple and robust. The approach consists of obtaining slant information from GP of the word image, and performing image processing on the GP. Words have been taken from the IAM Database of handwritten words. For slant correction, Interpolation Method has been used. An SVM-based OCR engine has been used to evaluate the efficacy of the proposed approach. The performance of the proposed approach has been compared to contemporary techniques. The results show that while the proposed approach performs better than the uniform slant correction approach, it performs as well as the widely-used DP-based non-uniform slant correction approach, given the fact that it is simple and robust. Future work will include extending the novel technique to HMM-based OCR systems, and also for Arabic text OCR which is the subject of extensive research.

REFERENCES

[1] I. Ahmad and G. A. Fink, "Multi-stage HMM based Arabic text recognition with rescoring," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, 2015, pp. 751-755.

[2] A. Maqqor, A. Halli, K. Satori, and H. Tairi, "Using HMM Toolkit (HTK) for recognition of arabic manuscripts characters," in *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, 2014, pp. 475-479.

[3] S. Uchida, E. Taira, and H. Sakoe, "Nonuniform slant correction using dynamic programming," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, 2001, pp. 434-438.

[4] A. Kuhn, "Using Local Slant Correction to Normalize Handwritten Text Samples," 2005.

[5] R. Bertolami, S. Uchida, M. Zimmermann, and H. Bunke, "Non-Uniform Slant Correction for Handwritten Text Line Recognition," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 2007, pp. 18-22.

[6] M. Ziaratban and K. Faez, "Non-uniform slant estimation and correction for Farsi/Arabic handwritten words," *IJDAR*, vol. 12, pp. 249-267, 2009.

[7] R. Al-Hajj, L. Likforman-Sulem, and C. Mokbel, "Combining Slanted-Frame Classifiers for Improved HMMBased Arabic Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1165-1177, 2009.

[8] M. A. Ali, "Slant correction techniques based on Wigner-Ville distribution applied for Arabic handwritten characters recognition systems," in *Communications, Computers and Applications (MIC-CCA), 2012 Mosharaka International Conference on*, 2012, pp. 165-168.

[9] J. Das Gupta and B. Chanda, "Novel methods for slope and slant correction of off-line handwritten text word," in *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on*, 2012, pp. 295-298.

[10] F. Nadi, J. Sadri, and A. Foroozandeh, "A novel method for slant correction of Persian handwritten digits and words," in *Pattern Recognition and Image Analysis (PRIA), 2013 First Iranian Conference on*, 2013, pp. 1-7.

[11] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis, "Slant estimation algorithm for OCR systems," *Pattern Recognition*, vol. 34, pp. 2515-2522, 2001.

[12] G. Nicchiotti and C. Scagliola, "Generalised projections: a tool for cursive handwriting normalisation," in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, 1999, pp. 729-732.

[13] U. Marti and H. Bunke, "The IAM-database: An English Sentence Database for Off-line Handwriting Recognition," *IJDAR*, vol. 5, pp. 39-46, 2002.

[14] E. Kavallieratou, N. Fakotakis, and G. K. Kokkinakis, "Handwritten Character Recognition Based on Structural Characteristics," in *ICPR*, 2002, pp. 139-142.

[15] C. Cortes and V. Vapnik, "Support-vector network," *Machine Learning*, vol. 20, pp. 273-297, 1995.

[16] T. Fletcher. (2009), Support Vector Machines Explained.

[17] L. Shen-Wei and W. Hsien-Chu, "Effective multiple-features extraction for off-line SVM-based handwritten numeral recognition," in *Information Security and Intelligence Control (ISIC), 2012 International Conference on*, 2012, pp. 194-197.

[18] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1-27, 2011.

[19] L. Hailong and D. Xiaoqing, "Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, 2005, pp. 19-23 Vol. 1.

[20] L. Cheng-Lin, "Normalization-Cooperated Gradient Feature Extraction for Handwritten Character Recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, pp. 1465-1469, 2007.

[21] A. M. Hafiz and G. M. Bhat, "Effect of Intrinsic Parameters on Efficiency of Gradient Features," *International Journal of Imaging and Robotics*, vol. 16, pp. 35-44, 2016.